AIAA Modeling and Simulation Technologies Conference and Exhibit
18 - 21 August 2008, Honolulu, Hawaii

AIAA 2008-7101

AF2008102

# Automating the Process of Terminal Area Node-Link Model Generation

Hak-Tae Lee *
*University of California Santa Cruz, Moffett Field, CA 94035*
Thomas F. Romer †
*NASA Ames Research Center, Moffett Field, CA 94035*

**The demand for detailed airport surface and terminal airspace models is growing as the scope and fidelity of air transportation system simulations expand. Currently, there are few detailed airport surface models used in air transportation system simulation and analysis, because the models are complex to develop and maintain. An improved method of generating detailed models is required to expand the set of detailed airport surface and terminal airspace models. In this paper, a means to automate the development of terminal area node-link graph models is investigated. Geographic information system data were processed through the geometric algorithms developed to capture the layout and connectivity of runways and taxiways, and node-link graphs were created using this information. The procedures were tested on about 80 US airports and proved to be robust and accurate.**

## Nomenclature

| | |
|---|---|
| $AR$ | Effective aspect ratio |
| $I$ | Two-dimensional area-based moment of inertia, [m$^4$] |
| $N$ | Number of vertices |
| $R$ | Earth radius, [m] |
| $X, Y$ | Centroid centered coordinates |
| $\mathbf{C}$ | Connectivity matrix |
| $\mathbf{I}$ | Moment of inertia tensor, [m$^4$] |
| $\mathbf{q}$ | Potential field induced by a unit vortex |
| $\mathbf{r}$ | Position vector |
| $\mathbf{v}$ | Principal direction, eigenvector |
| $c_{ij}^s$ | Element of $\mathbf{C}^2$ |
| $d$ | Dimensional tolerance, [m] |
| $x$, $y$ | Coordinates |

*Subscripts*

| | |
|---|---|
| $1, 2$ | Along major and minor principal axes |
| $x$, $y$ | Along $x$ and $y$ axes |
| cen | Centroid |

*Conventions*

| | |
|---|---|
| N$i$ | Node with index $i$ |
| P$i$ | Polygon with index $i$ |

*Symbols*

| | |
|---|---|
| $\alpha$,$\beta$ | Interpolation weights |

*Aviation Systems Research Engineer, Aerospace Operations Modeling Branch, M/S 210-8, Moffett Field, CA 04035, AIAA Member

†Aerospace Engineer, Wind Tunnel Operations Branch, M/S 227-5, Moffett Field, CA 94035

$\epsilon_1$ Non-dimensional tolerance in perpendicular direction
$\epsilon_2$ Non-dimensional tolerance in line segment direction
$\bar{\lambda}$ Reference latitude, [rad]
$\lambda$ Latitude, [rad]
$\phi$ Potential
$\tau$ Longitude, [rad]

## I. Introduction

Most future air traffic concepts involve operations at higher terminal airspace and airport surface densities to meet the expected growth in demand.[1] Air transportation system simulations involving detailed terminal areas and surfaces are necessary to assess these future concepts. These simulations will include studies of individual airports[2] as well as regional studies.[3] All of them will require detailed surface and terminal airspace models representing both current and future operating configurations. Because no two airports are identical, proper analysis of a concept's capabilities and benefits will require modeling operations at many airports.

To simulate the movements of aircraft on the airport surface, it is convenient to use node-link graph models, since the movements are constrained by the taxiways and runways. Modeling aircraft movements on a node-link model makes it straightforward to determine separation between aircraft. Another benefit of graph representation is that various depths of detail can be modeled depending on the simulation requirements.[3]

Developing, maintaining, and modifying even a small number of detailed terminal area node-link models are significant tasks. Most existing detailed surface models are created manually by extracting the coordinates of nodes from data sources such as aerial photography and connecting adjacent nodes with links. Atkins et al.[4] describe in detail such a manual model generation procedure for a surface simulation and decision support tool.[2]

One of the few efforts to automate the surface node-link model generation process is found in Ref. 5. In this work, a feature from the geographic information system data[4,6] that represents the actual taxiway centerlines is used to generate node-link models. Although this approach can result in accurate node-link graphs, the taxiway centerline feature is not available for many airports, limiting the number of airports that can be quickly modeled.

In the present study, procedures that automatically generate node-link graphs for airport surfaces are developed. Geographic information system data of airports, mainly composed of polygons, are processed through a set of geometric algorithms developed in this study. These algorithms capture the layout and connectivity of runways and taxiways, which are the two primary elements of airport infrastructure. They were used to successfully create node-link models for 77 of 78 airports in the data set.

The structure of this paper is as follows. In Section II, source data are described. Section III explains the step-by-step procedures taken to automatically generate the surface node-link graph. Section IV shows the automatically generated node-link graph of selected airports to demonstrate the effectiveness of the procedures developed in this study and discusses issues and concerns. Section V concludes the paper. The geometric algorithms are listed in the Appendix.

## II. Airport Surface Information

Safe Flight 21 (SF21) is a program sponsored by the Federal Aviation Administration (FAA) to explore and demonstrate the use of Automatic Dependent Surveillence-Broadcast (ADS-B) and other related technologies for use in the free flight concept of operations.[6] As part of the SF21 project, the FAA created digitized airport maps by processing aerial photography for about eighty airports to facilitate airport surface automation.[4]

Among several available digitized map data, the SF21 data was chosen for the present study mainly because many of the existing surface simulation and decision support tools[2] are based on this data, and the entire data set was readily available to NASA. Moreover, other commercial data sets such as Jeppesen Geo Spatial Information Technology data were determined not to provide any notable advantage over the SF21 data for the purpose of this effort.

Safe Flight 21 data are a collection of polygons or lines categorized by different fields such as runway, taxiway segment, taxiway guidance line, and other fields. Each polygon is represented by a list of latitude

and longitude pairs for the vertices that define the polygon. Figure 1 shows the runway and taxiway polygon representation of San Francisco International Airport (SFO).

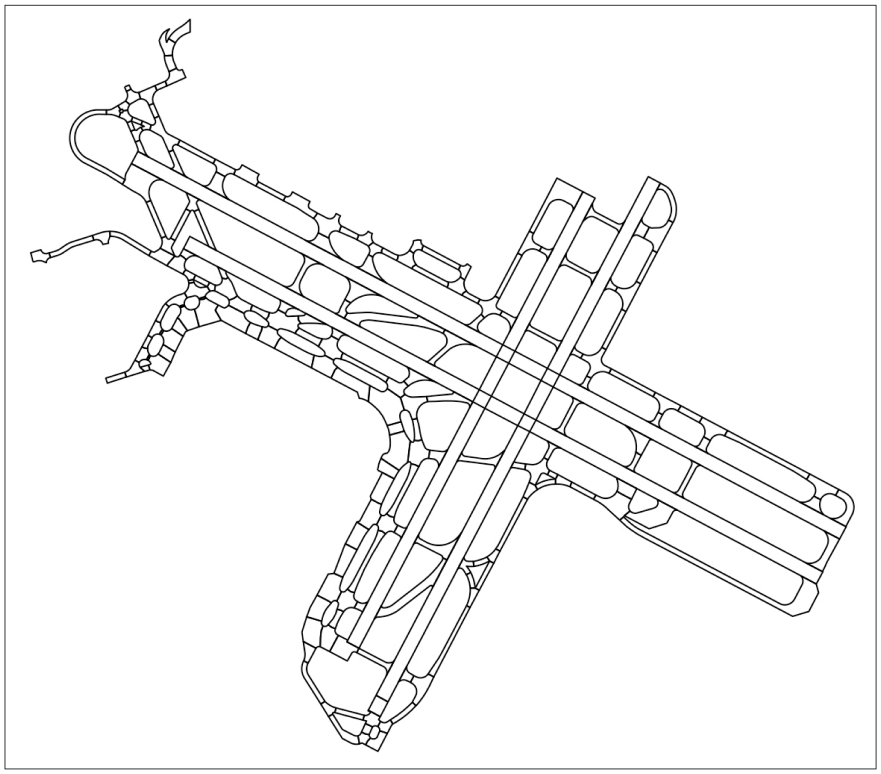


**Figure 1. Runway and taxiway polygons for San Francisco International Airport.**

## III. Procedures for Automated Node-Link Model Generation

Detailed procedures for automated node-link model generation are described throughout the following subsections. The input data are pre-processed for better conditioning and defect corrections. Then, taxiway polygons are processed for connectivity, and runway polygons are processed to find runway center lines. Next, runway entry and exit nodes and links are created to construct the baseline graph. Finally, the baseline graph is post-processed to adjust the node-link placements and to remove unnecessary node-links.

The core of the automated node-link generation process is finding connectivity between polygons. Two polygons are considered to be connected if the polygons share a series of line segments whose total length is longer than a predetermined threshold. These shared lines will be referred as shared borders in the following sections. If the two polygons are connected, a node is created at the centroid of each polygon, and the two nodes are connected by a link as depicted in Fig. 2.

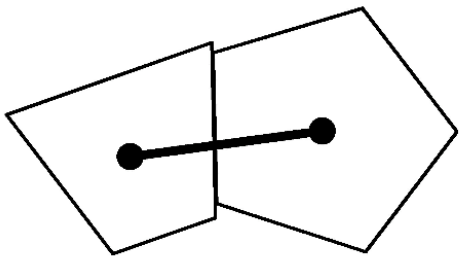
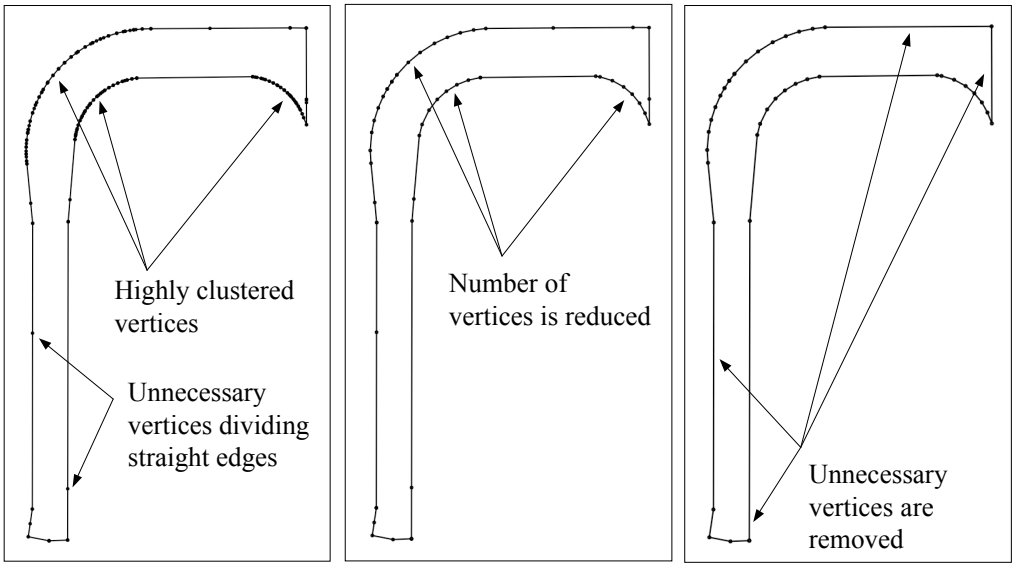


**Figure 2. Two polygons are connected.**

### III.A. Pre-processing

The original input data contain various defects that cause many geometric algorithms to fail or run very slowly. By pre-processing the data, significant portions of these problems can be removed.

Each polygon is, first, examined for duplicated vertices, which result in zero-length edges. If duplicated vertices are found from the vertex list, they are reduced to a single vertex.




(a) Original polygon. (b) Length filter is applied. (c) Angle filter is applied to (b).

**Figure 3. Pre-processing.**

Then, the polygon goes through an edge length filter, which constructs a new polygon by selectively adding vertices from the original polygon. Vertices are selected such that the distance between any two consecutive vertices is larger than the given minimum edge length. The original polygon is replaced by the new polygon which, generally, has fewer vertices. Figure 3 (b) shows the result of the edge length filter applied to the original polygon shown in Fig. 3 (a), displaying a significant reduction in the number of vertices.

After the length filter is applied, an angle filter is applied to remove vertices which have an exterior angle that is smaller than the threshold. This filter is used to remove unnecessary vertices that divide straight edges into shorter edges. Figure 3 (c) shows the result of the angle filter applied to Fig. 3 (b).

The length and angle filters effectively reduce the number of vertices for faster processing, while retaining the basic shape of the polygon. Threshold values are empirically determined to balance between vertex reduction and shape retention. For the given data, 7 meters is used for the minimum edge length, and 1.5 degrees is used for the minimum exterior angle.

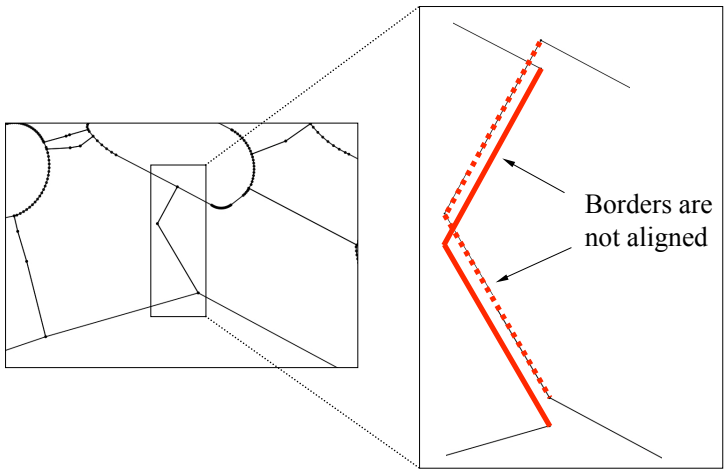


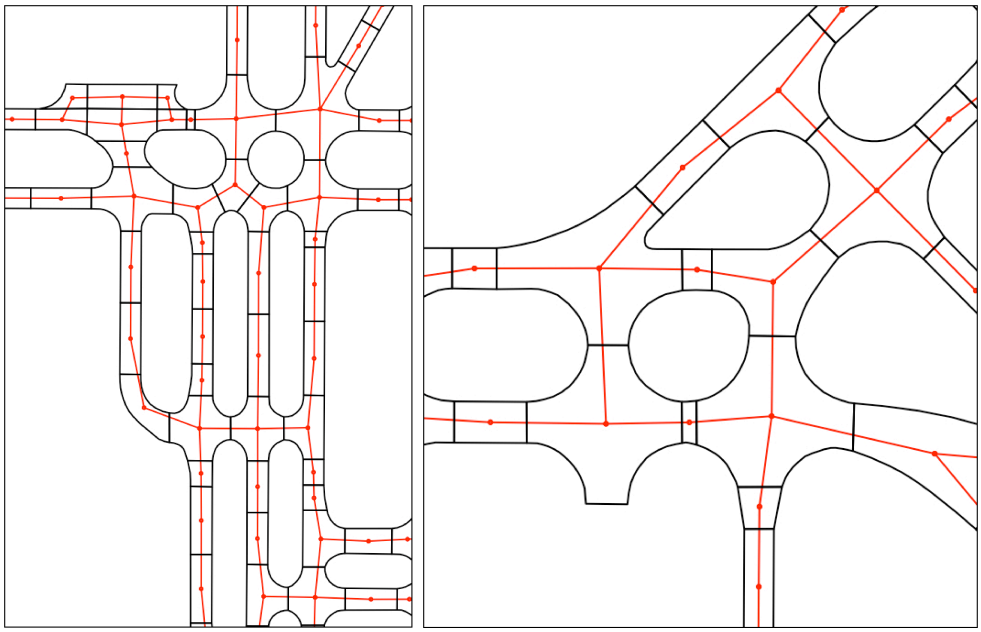

**Figure 4. Example of misalignment.**

After all the polygons are conditioned, shared borders between polygons are examined to treat mis-

alignments such as those shown in Fig. 4. As the connectivity algorithm searches for the shared borders between two polygons, misalignment can cause the algorithm to miss the connectivity. Automatically treating misalignment is important because it is very difficult to detect the misalignment by visually inspecting a graphical display. The neighborhood of each vertex is searched for other vertices. Vertices that are found to be within a certain distance are consolidated to a single vertex.

### III.B. Creating Node-Links for Taxiways

Figure 5 shows two examples of taxiway node-links generated by placing nodes at the centroids and connecting them by links if the polygons are connected. Although the links do not exactly follow the centerlines of the taxiways, the connectivity is correctly captured. As can be seen from Fig. 5, some of the nodes are unnecessary. These nodes will be removed during the post-processing step later.

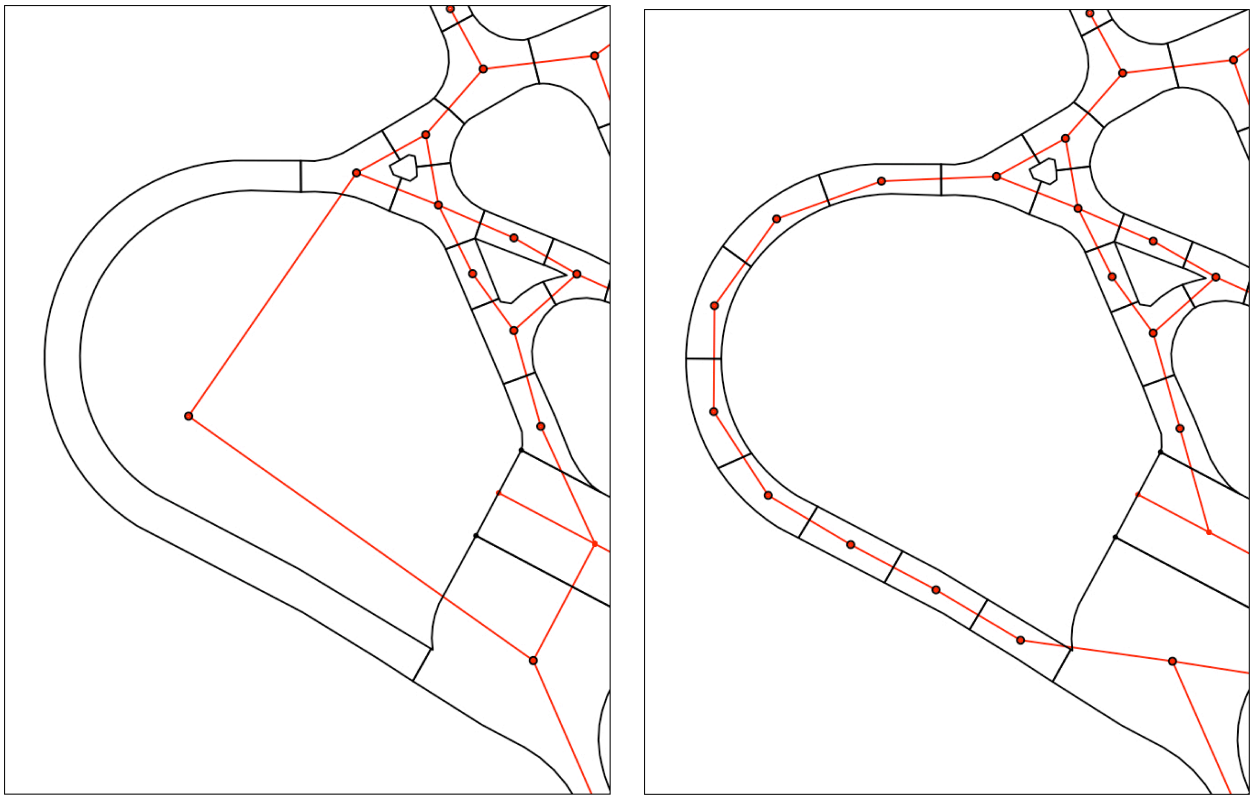


(a) (b)

**Figure 5. Examples of taxiway node-links.**

Some of the polygons are long and narrow, and often concave. In this case, putting a single node at the centroid of the polygon can create node-links which significantly deviate from the actual taxi paths. Figure 6 (a) shows links that go out of the taxiway boundary. To mitigate this problem, polygons are divided until they satisfy two criteria. The first criterion is that the effective aspect ratio of the polygon must be smaller than the threshold. Effective aspect ratio is defined as the aspect ratio of the effective rectangle, which has edge lengths and orientation such that its principal directions and the moment of inertias coincide with those of the original polygon. The second criterion is called an interior condition, which means that the centroid should be inside the polygon. Any polygon that does not meet both criteria is divided along the principal axis that is associated with the largest moment of inertia. Figure 6 (b) shows an example of polygon division and the resulting node-links.

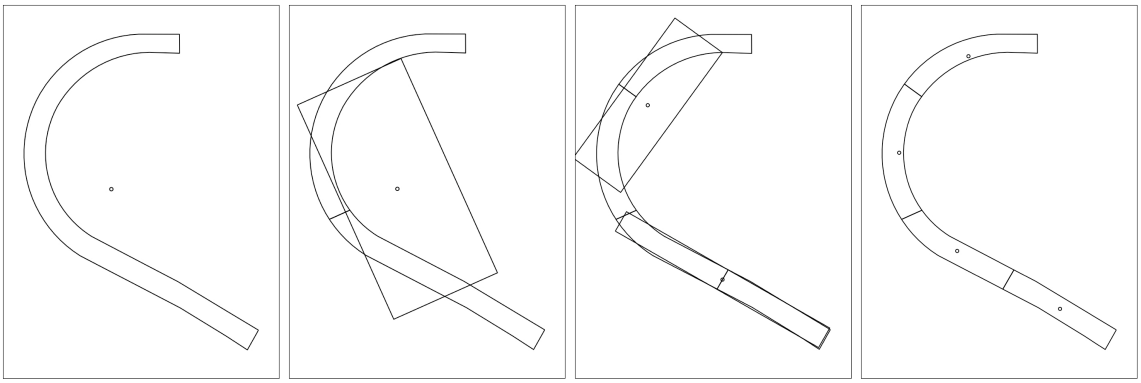
Figure 7 describes the division process in detail. Figure 7 (a) shows a J-shaped polygon that does not meet the interior condition. Figure 7 (b) shows it divided into two parts, along with its effective rectangle. Figure 7 (c) shows the upper portion being divided because it violates the interior condition and the lower part being divided because its effective aspect ratio is too large. Figure 7 (d) shows the original polygon divided into four polygons. In real world example shown in Fig. 6, maximum effective aspect ratio is 4, and the original polygon is divided into eight polygons.



(a) Before division.

(b) After division.

**Figure 6. Polygons are divided to make the links follow the shape of the taxiway more closely.**



(a) J-shaped polygon and its centroid.

(b) Divided into 2 polygons with effective rectangle overlaid.

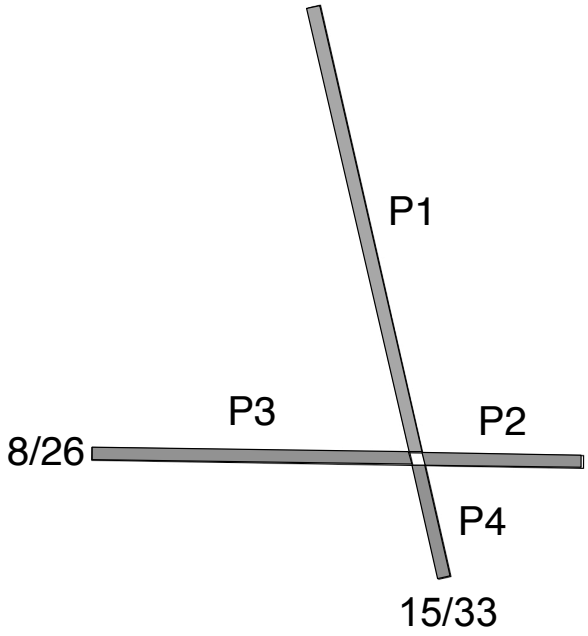
(c) Divided again.

(d) Divided into 4 polygons.

**Figure 7. Polygons division process.**

### III.C. Identifying and Grouping Runways

The source data represent runways using single or multiple polygons. If a runway does not have any other crossing runways, it is represented by a single rectangular polygon. However, if there are crossing runways, two or more polygons are used to represent a runway, and these polygons are generally not rectangular. Figure 8 shows the runway portion of the input data for Burbank, California's Bob Hope Airport (BUR). Polygon P1 and P4 represent Runway 15/33, and polygon P2 and P3 represent Runway 8/26.




$$\mathbf{C} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

$$\mathbf{C}^2 = \begin{bmatrix} 2 & 0 & 0 & 2 \\ 0 & 2 & 2 & 0 \\ 0 & 2 & 2 & 0 \\ 2 & 0 & 0 & 2 \end{bmatrix}$$

**Figure 8. Example of crossing runways at Burbank, California's Bob Hope Airport, (BUR).**

To correctly group these polygons, a connectivity matrix, $\mathbf{C}$, is constructed by the rule described in Eq. 1. Runway polygons are considered to be connected if they share a vertex. This less strict connectivity condition is used only for processing runways. For example, from the runway configuration shown in Fig. 8, polygon P1 is connected to polygon P2 and P3. The completed connectivity matrix is shown in Fig. 8.

$$\mathbf{C}_{ij} = \begin{cases} 0 & \text{if } i = j \text{ or P}i \text{ and P}j \text{ are not connected} \\ 1 & \text{if P}i \text{ and P}j \text{ are connected} \end{cases} \tag{1}$$

Each element, $c^s_{ij}$, of the square of the connectivity matrix, $\mathbf{C}^2$, indicates the number of polygons that are connected to both polygon P$i$ and polygon P$j$. An element along the diagonal of $\mathbf{C}^2$, $c^s_{ii}$, represents the total number of polygons connected to polygon P$i$. For example, if $c^s_{ii}$ is zero, polygon P$i$ is a stand alone runway.

To further process crossing runways such as the layout shown in Fig. 9 (a), a temporary node-link graph, as shown in Fig. 9 (b), is created. A node, N$i$, is placed at the centroid of runway polygon, P$i$. Two nodes, N$i$ and N$j$, are connected by a link only if $c^s_{ij}$ is equal to two, which means there are two polygons that are connected to both polygon P$i$ and polygon P$j$. This graph is used only for runway identification, thus it is not related to the airport surface node-link graph.

Once the graph is constructed, the rank of each node is counted. Rank of a node refers to the number of links attached to this node. If a node has a rank higher than two, all the link pairs are examined to find the pair that forms a straight line. In the example shown in Fig. 9 (c), N4 has a rank of three. Among the three possible link pairs, the link connecting N3 and N4, and the link connecting N4 and N5 form a straight line. After this pair is found, other links are removed. In the example, the link that is connecting N4 and N9 is removed. This procedure is repeated until none of the nodes has a rank greater than two, as shown in Fig. 9 (e).

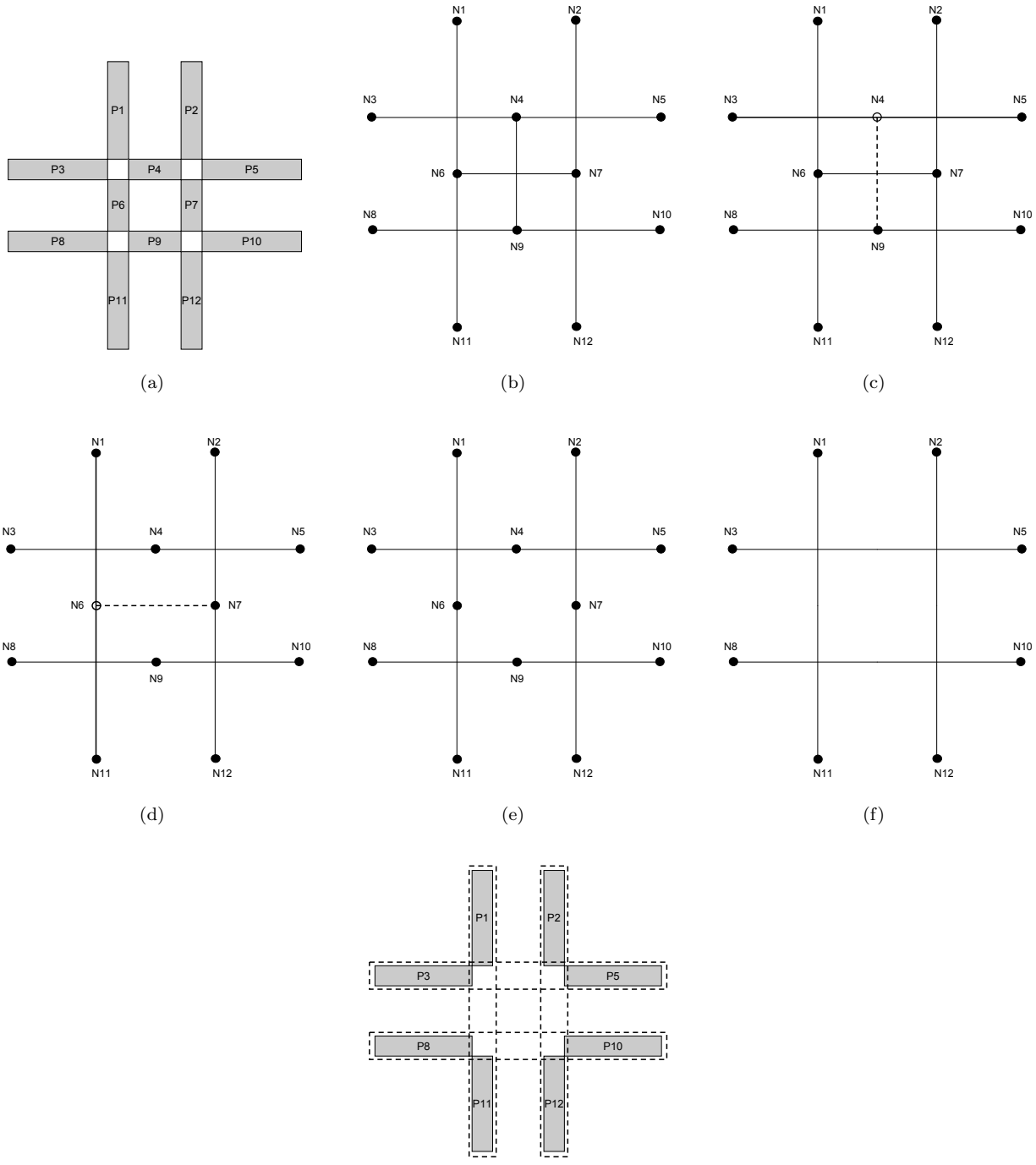
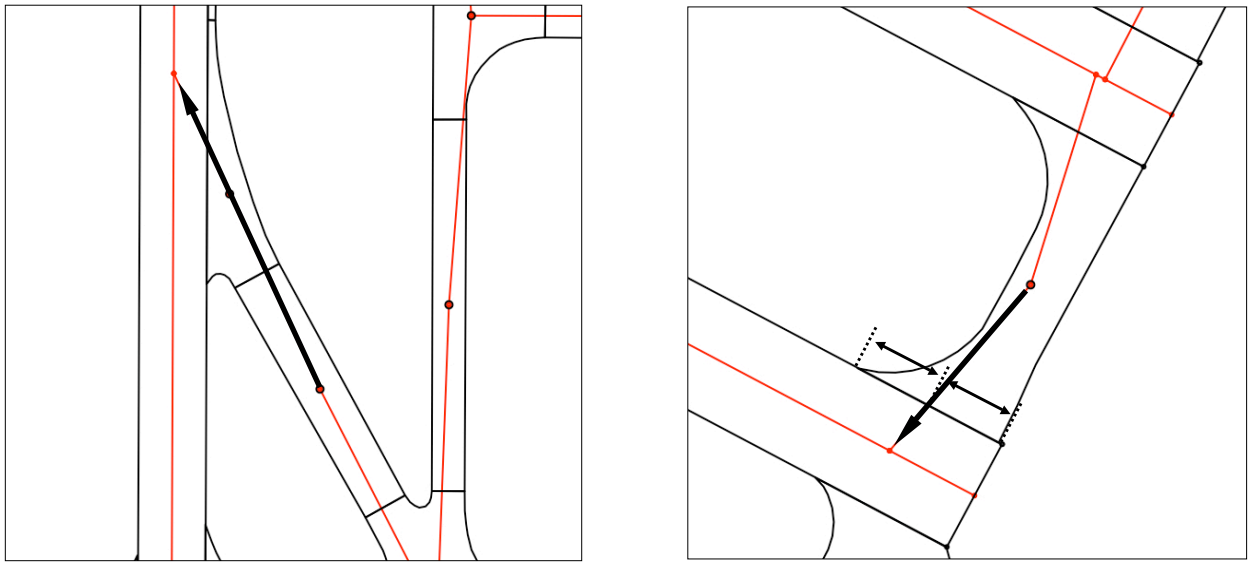



**Figure 9. Runway processes.**

The process separates the graph into a number of straight line graphs, and each graph represents a runway. In the example shown in Fig. 9 (e), initial graph of Fig. 9 (b) is separated into four straight line graphs. From the straight line graph, two end nodes are identified as shown in Fig. 9 (f), and, finally, two polygons corresponding to the end nodes are wrapped around by a single rectangle as shown in Fig. 9 (g). Runway centerline is identified for each runway using this rectangle, and nodes are placed at the runway crossings.

### III.D. Connecting Runways and Taxiways

After the taxiway node-link graph is constructed and all the runway centerlines are identified, runway entry and exit nodes along the runway centerlines are generated. First, a taxiway polygon that is connected to a runway polygon is identified. Then, the link ending at this taxiway polygon is extended towards the runway centerline as shown in Fig. 10 (a). A node is placed at the intersection of the extension and centerline. This method, called the primary method, is desirable because it generally preserves the runway entry and exit angle. However, if only a single taxiway polygon is adjacent to a runway polygon or the link extension of the primary method does not cross the runway centerline, a secondary method is applied. Connection to the runway is made by creating a link from the centroid that goes through the mid-point of the shared border as shown in Fig. 10 (b).



(a) Primary procedure (b) Secondary procedure

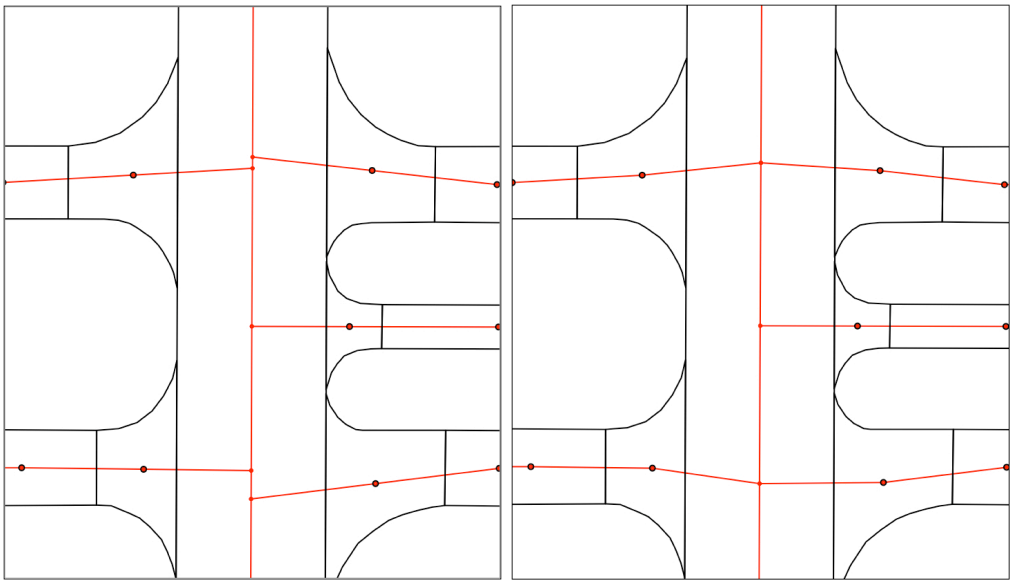
**Figure 10. Creating runway entry and exit links.**

As shown in Fig. 10 (b) and Fig. 11 (a), if a runway has entry and exit to both sides, two nodes are generated along the runway centerline for each entry and exit. These nodes should be consolidated to a single node as shown in Fig. 11 (b). Node consolidation is performed by checking the distance between every pair of nodes along the runway centerline. If the distance is smaller than 50 meters, the two nodes are consolidated to a single node by taking the mid point.

### III.E. Post-processing

The node-link graphs created using the previously mentioned procedures capture the layout and connectivity of taxiways and runways. However, they can be further improved by post-processing. First, some of the nodes will be relocated from their initial location at the centroid to make the links follow the taxiway centerlines more closely. Next, intermediate nodes will be removed to make the graph simpler without losing any essential information.

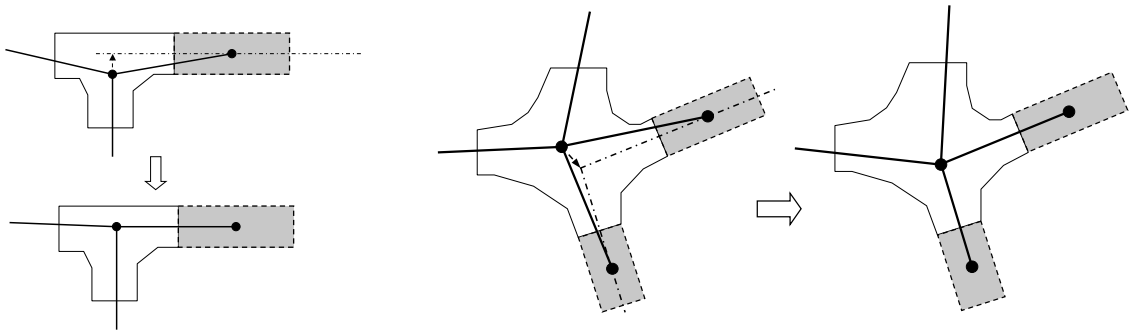
Nodes are relocated using the readily identifiable centerlines of the rectangular polygons. First, each polygon is examined to determine if it is a rectangle. Next, for every non-rectangular polygon, the number of connected rectangular polygons is counted. If no rectangle is connected to this non-rectangular polygon,



(a) Initial linking between taxiway and runway (b) Runway entry/exit nodes are consolidated

**Figure 11. Consolidation.**

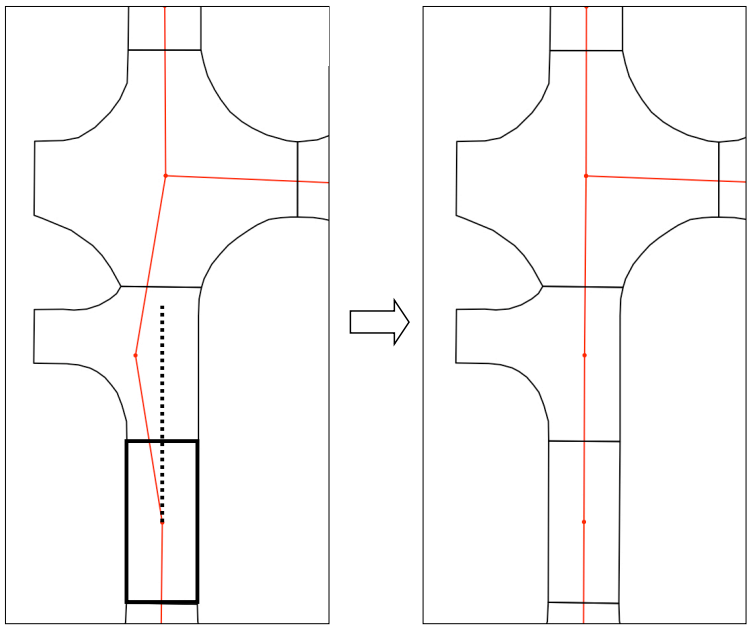
the node remains at the centroid. If one connected polygon is rectangular, the centerline of the rectangle is extended towards this polygon and the node is projected to this line as shown in Fig. 12 (a). If two or more rectangles are connected, crossing angles between the extended centerlines are measured for every possible pair, and the pair that forms an angle closest to 90 degrees is selected. This crossing point becomes the new node location for the polygon as shown in Fig. 12 (b). Figure 13 shows real world examples of rectangle-based correction.



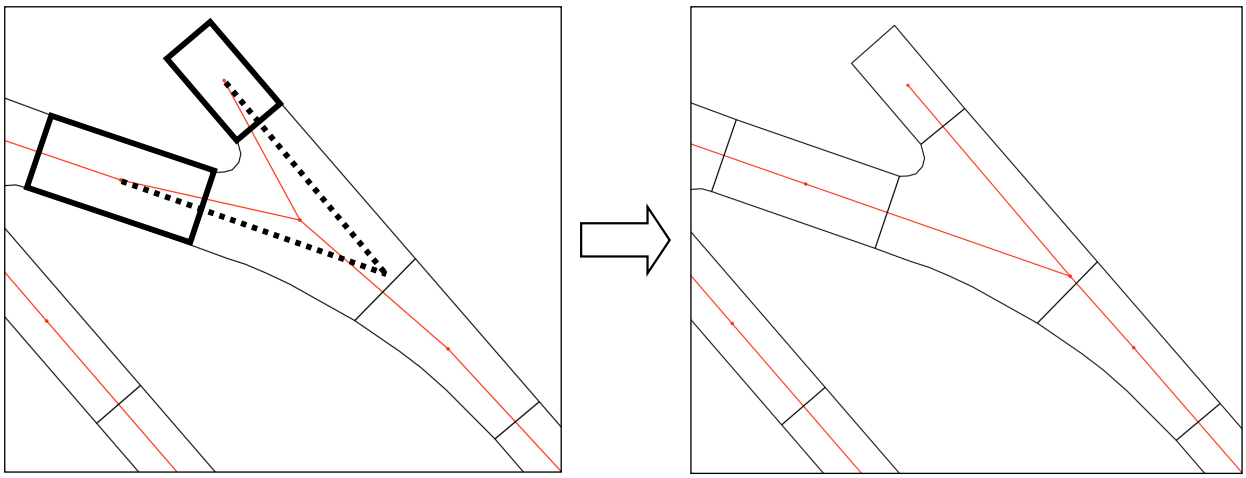
(a) Only one adjacent polygon is a rectangle. (b) Two or more adjacent polygons are rectangles.

**Figure 12. Post-processing: Rectangle based correction.**

After the node positions are adjusted using the rectangle-based correction, the graph is simplified by removing unnecessary nodes. If a node has only two connected links, and the exterior angle between the links is smaller than a given tolerance, this node is removed and the two links are replaced with a single link. An example of graph simplification for a typical airport is shown in Fig. 14. A tolerance angle of 15 degrees is used for the actual computation.

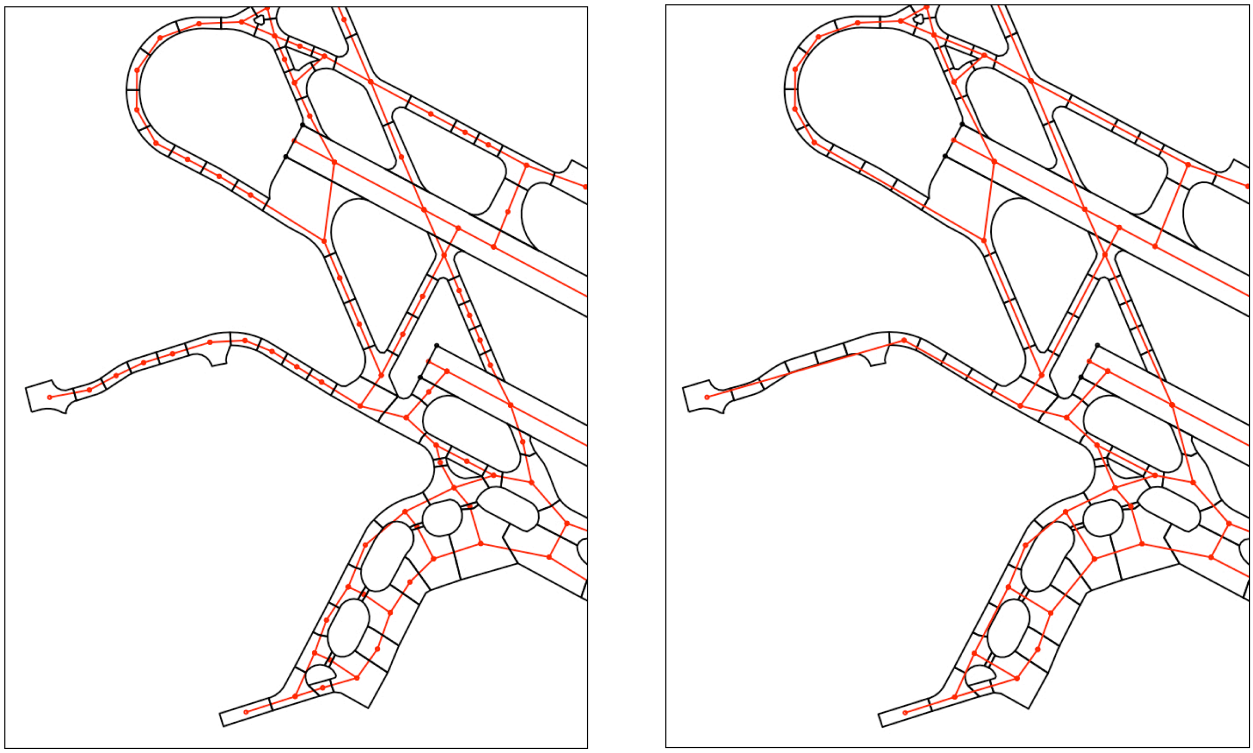


(a)



(b)

**Figure 13. Rectangle based correction examples.**



(a) Before simplification (b) After simplification

**Figure 14. Post-processing: Graph simplification by removing intermediate nodes.**

## IV. Results

The SF21 source data, provided 78 airport geometries. Node-link graphs were created for 77 airports by applying the processes developed in this study. The remaining one airport that failed is discussed later in this section. Among the 77 airports, three representative ones, Dallas Fort Worth International Airport (DFW), Chicago O'Hare International Airport (ORD), and San Francisco International Airport (SFO) are presented in this section.

Figure 15 shows the result for DFW. Red dots and lines denote nodes and links, while black lines denote runway and taxiway outlines as described by the source data. This airport was chosen as the primary test case because it has one of the most complicated taxiway networks among the US airports, and many of NASA's new air traffic concepts are tested on DFW. The data for DFW required very few corrections through pre-processing. Moreover, the well structured taxiway layout enabled the rectangle based correction algorithm to perform effectively.

Figure 16 shows the node-link graph for Chicago O'Hare International Airport (ORD). Contrary to DFW, which is relatively new and designed with generous space, ORD represents a relatively old and complex airport with limited space. While processing ORD, the moment-of-inertia based polygon division algorithm was particularly useful, since many of the taxiway polygons were long, narrow, and concave.

San Francisco International Airport (SFO) is of medium complexity. However, the original data contains many defects such as misalignments. San Francisco International Airport served as the main test case for the runway processing algorithm since it has two pairs of parallel runways crossing each other. Figure 17 shows the result for SFO, which indicates that all the runways are correctly recognized and the defects were effectively removed through the pre-processing.

There are two major concerns with the methods described in this study. One is related to the source data, and the other is having many empirically determined tolerance values.

Although the processes generated accurate node-link graphs of 77 airports, the graphs are only as good as the input data. The SF21 data were last updated in 2003, so any changes in the airport layouts after 2003 are not reflected. A separate tool that can manually edit the graph would be useful to make updates.

The quality of the node-link graphs depends on the quality and the consistency of the input data. For

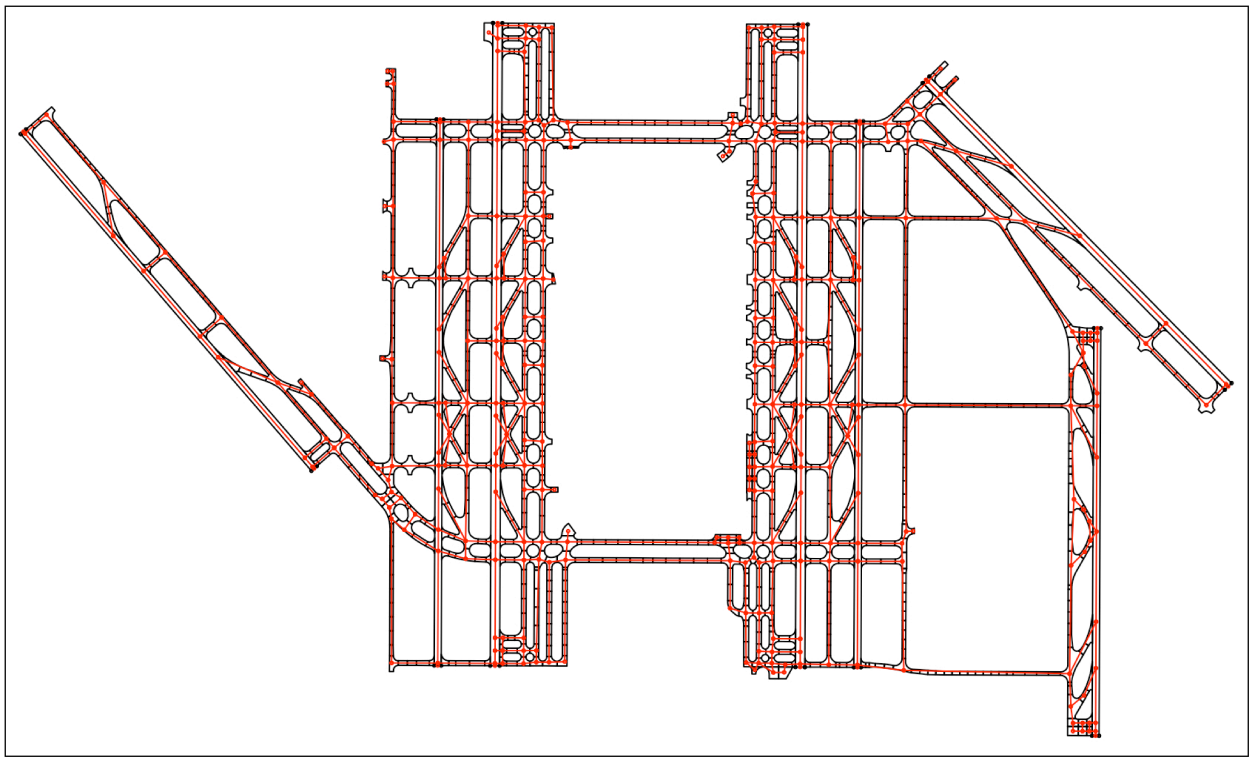


**Figure 15. Node-link graph of Dallas Fort Worth International Airport.**

example, node-link graphs could not be generated automatically for Albuquerque International Sunport Airport (ABQ). It has three runways crossing at a single point as shown in the circled area of Fig. 18. The runway processing algorithm described in Section III breaks down with this case.

As mentioned throughout Section III, numerous threshold values were introduced. These values are highly dependent on the input data, so they were empirically determined for the given SF21 data. These values should be adjusted accordingly if any other geographic information system data should be used.

## V. Conclusions

In this study, automating the airport node-link graph model generation process was investigated. Geometric algorithms were developed to manipulate the polygons provided by the source geographic information systems data. Using these algorithms, node-link graphs of the airport surfaces were extracted by identifying polygon connectivity. Among the 78 airports provided by the source data, node-link graphs for 77 airports were created successfully. Resulting graphs accurately captured the connectivity and layouts of runways and taxiways, which demonstrated the robustness of the automated procedures.

Node-link graphs created from this study can be used as the foundation for creating complete models for terminal areas. Since the automation reduces the time and effort required for generating models, it will enable simulations and analyses involving many airports, as well as the interconnecting network.

## Appendix

In this appendix, geometric algorithms specifically developed for handling polygons and finding connectivity are described in detail. Most of the algorithms are based on classical plane geometry.

### Coordinate System and Projection

All of the airport surfaces were assumed to be locally flat within each airport. For each airport, runway polygons are searched to find maximum and minimum latitudes. The average of the two latitudes becomes

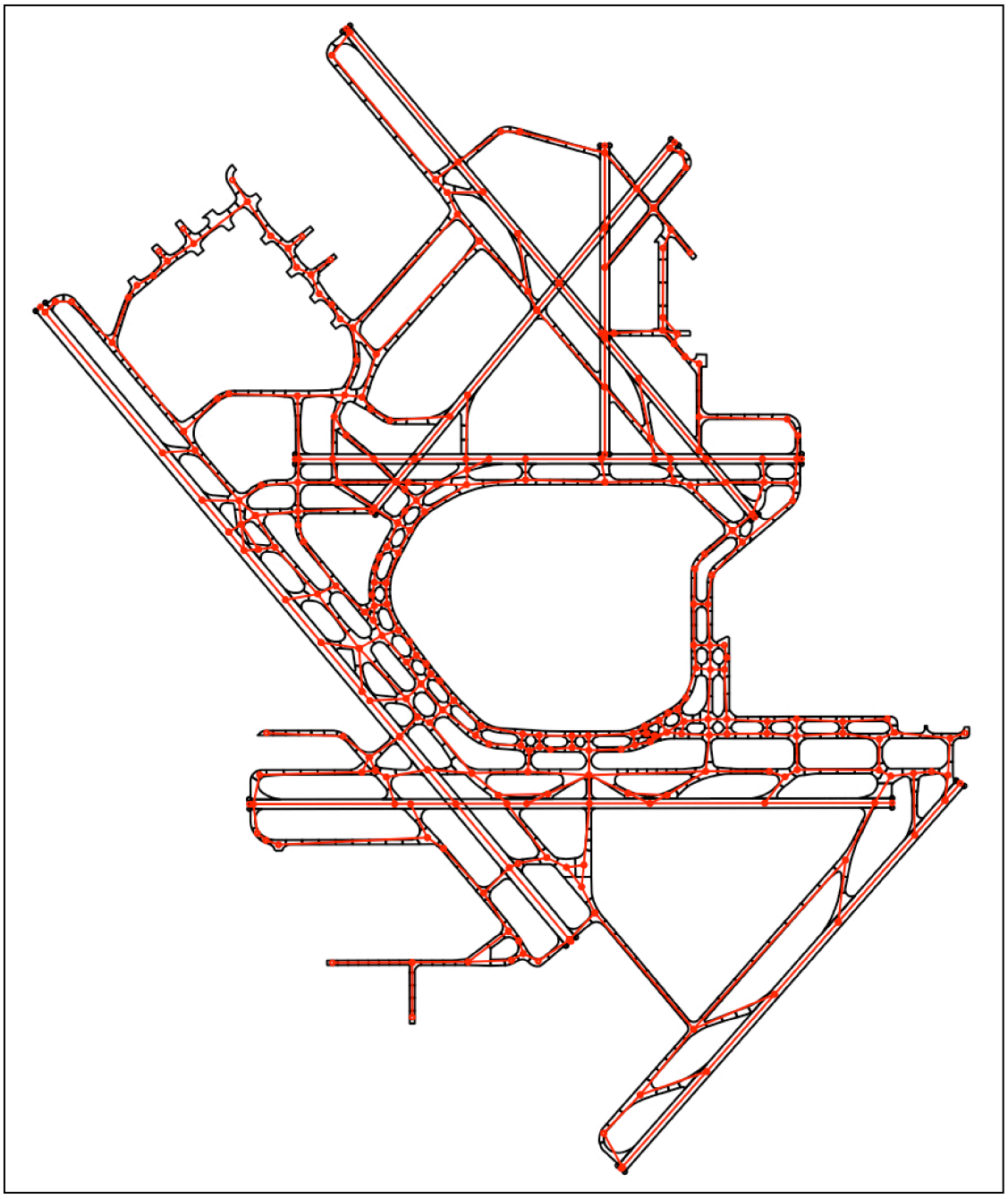


**Figure 16. Node-link graph of Chicago O'Hare International Airport.**

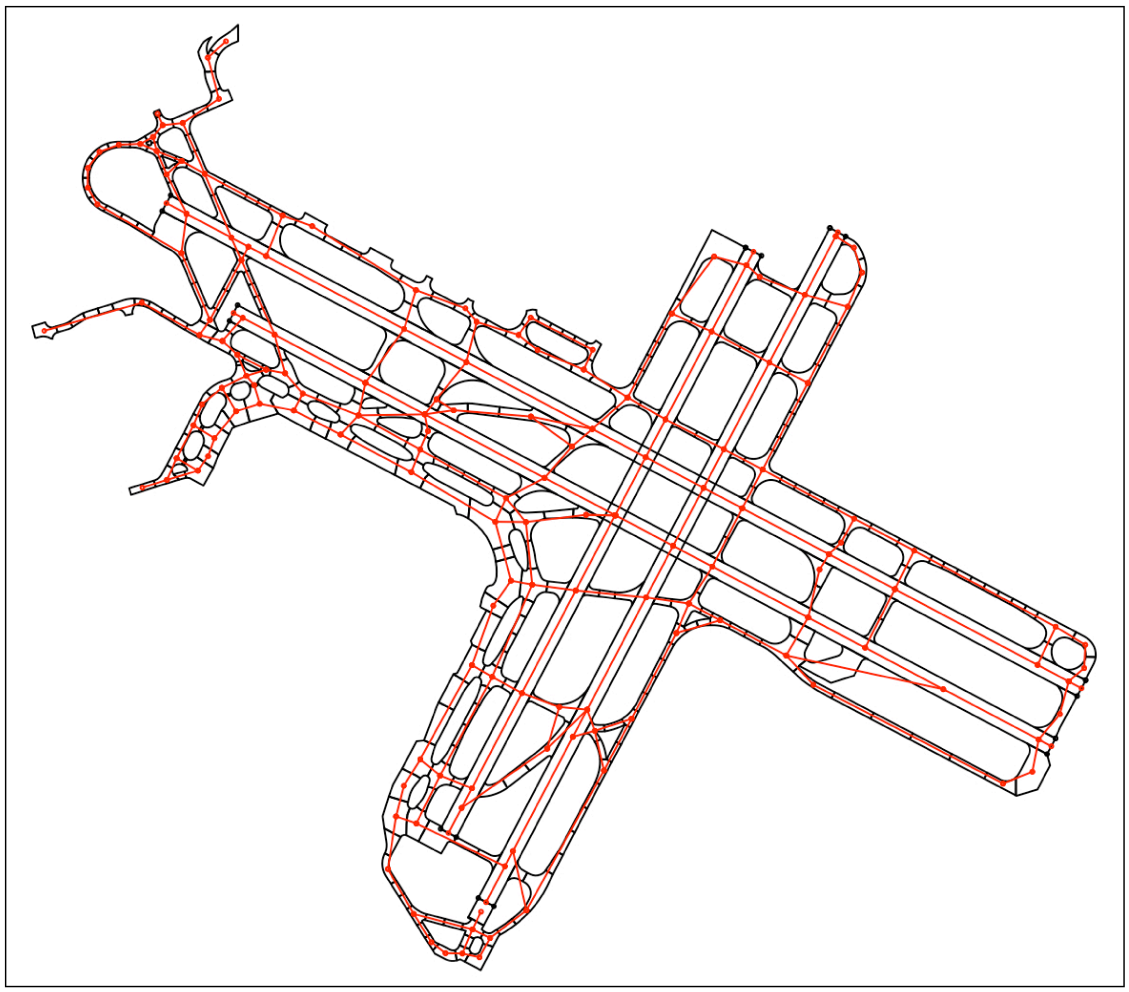


**Figure 17. Node-link graph of San Francisco International Airport.**

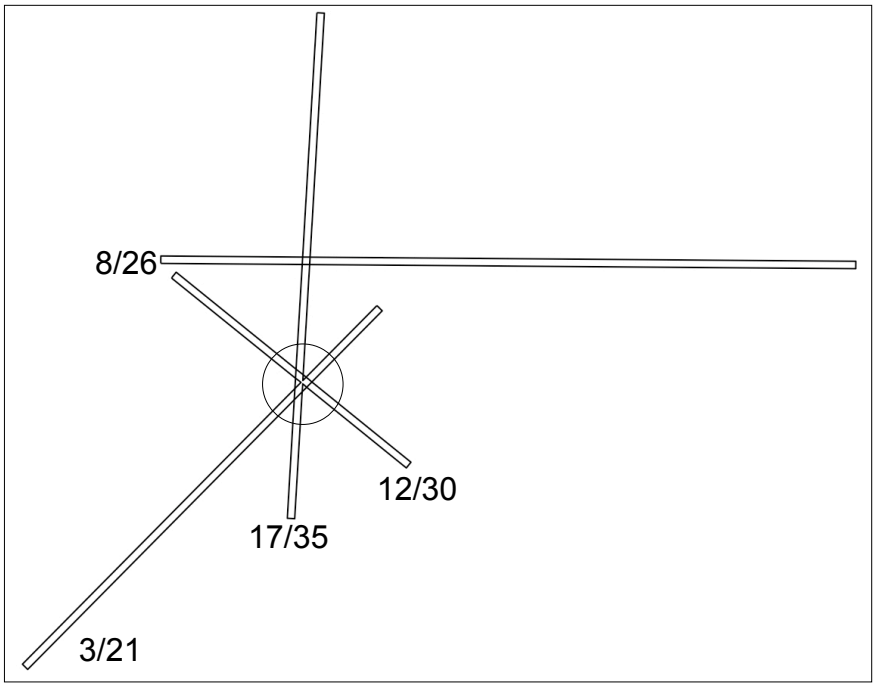



**Figure 18. Runways at Albuquerque International Sunport Airport. Runway 3/21, 12/30, and 17/35 meet at a single point.**

the reference latitude, $\bar{\lambda}$. All of the coordinates of vertices, which are originally expressed in latitude and longitude pairs, are converted to plane coordinates using Eq. 2. Positive $x$ direction points toward east and positive $y$ direction points towards north.

$$\mathbf{r} = (x, y) = \left(R\tau \cos\bar{\lambda}, R\lambda\right) \tag{2}$$

### Point and Line Segment Relation

Shared borders, which determines the polygon connectivity, are identified by testing all of the edges of one polygon against the edges of the other polygon. If any two edges share a line segment, it becomes a part of the shared border. Once all of these parts are identified, they are rearranged to form a series of line segments, the shared border. When testing a pair of edges, the end points of one edge is examined to determine if they lie on the other edge, and vice versa.

The geometric algorithm that determines if a point lies on a given line segment is the core algorithm, which eventually leads to the connectivity between polygons. Any point, $\mathbf{r}$, on a straight line going through two points, $\mathbf{r}_1$ and $\mathbf{r}_2$, can be expressed as Eq. 3. If $\alpha$ is between zero and one, $\mathbf{r}$ lies on the segment. However, this mathematical condition is seldom satisfied exactly in real world situations. It is necessary to specify a region around the line segment as shown in Fig. 19. If a point is inside this region, the point is considered to be on the given segment. For example, in Fig. 19, point $\mathbf{r}$ is on the segment while point $\mathbf{r}'$ is not.

$$\mathbf{r} = (1 - \alpha)\,\mathbf{r}_1 + \alpha\mathbf{r}_2 \tag{3}$$

Equation 3 is relaxed to Eq. 4 where the weights, $\alpha_1$, $\alpha_2$, and $\beta$ should satisfy the conditions given by Eq. 5 and 6. The geometric tolerance, $d$, is translated to algebraic tolerance, $\epsilon_1$ and $\epsilon_2$, by Eq. 7 and 8. The value of $d$ is set to 0.5 meters.

$$\mathbf{r} = \beta\left(\alpha_1\mathbf{r}_1 + \alpha_2\mathbf{r}_2\right) \tag{4}$$

$$|\alpha_1 + \alpha_2 - 1| < \epsilon_1 \tag{5}$$

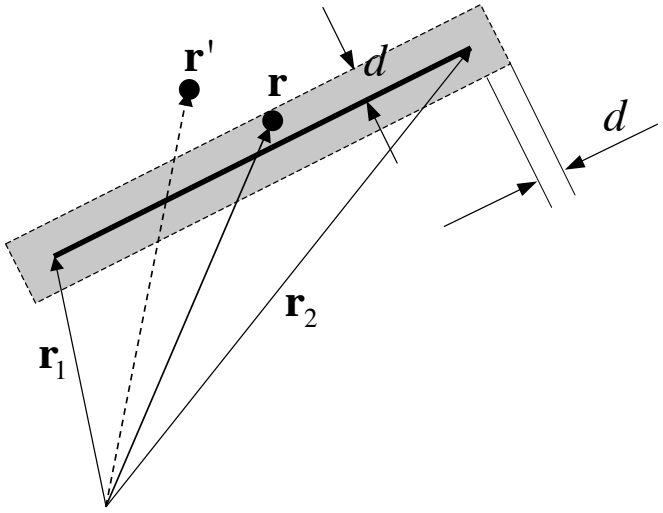



**Figure 19. Determining if a point is on a given segment.**

$$|\beta - 1| < \epsilon_2 \tag{6}$$

$$\epsilon_1 = d\frac{\|\mathbf{r}_2 - \mathbf{r}_1\|}{\|\mathbf{r}_1 \times \mathbf{r}_2\|} \tag{7}$$

$$\epsilon_2 = \frac{d}{\|\mathbf{r}_2 - \mathbf{r}_1\|} \tag{8}$$

where

$$\mathbf{r}_1 \times \mathbf{r}_2 = x_1 y_2 - x_2 y_1 \tag{9}$$

### Centroid

If a closed polygon as shown in Fig. 20 is expressed by a sequence of points, $\mathbf{r}_1$, $\mathbf{r}_2$, ... , $\mathbf{r}_N$, $\mathbf{r}_{N+1}$ where $\mathbf{r}_1$ overlaps $\mathbf{r}_{N+1}$, the centroid can be computed by Eq. 10. Equation 10 is particularly convenient because it does not matter whether the vertices are numbered in the clockwise direction or counterclockwise direction.

$$\mathbf{r}_{\text{cen}} = \frac{\sum_{i=1}^{N} (\mathbf{r}_i + \mathbf{r}_{i+1})(\mathbf{r}_i \times \mathbf{r}_{i+1})}{3\sum_{i=1}^{N} (\mathbf{r}_i \times \mathbf{r}_{i+1})} \tag{10}$$

### Moment of Inertia and Principal Axis

Principal axes for each polygon are computed to find effective aspect ratio and to divide polygons with high effective aspect ratios. Area-based moment of inertias are computed to find principal axes. First, each polygon should be translated such that the centroid becomes the origin as described in Eq. 11.

$$(X_i, Y_i) = \mathbf{r}_i - \mathbf{r}_{\text{cen}} = (x_i - x_{\text{cen}}, y_i - y_{\text{cen}}) \quad \text{where} \quad i = 1, 2, \ldots, N+1 \tag{11}$$

After the translation, moment of inertias are computed using Eqs. 12 through 14. Unlike the centroid, the sign of moment of inertias are dependent on the direction that the vertices are numbered. If the vertices

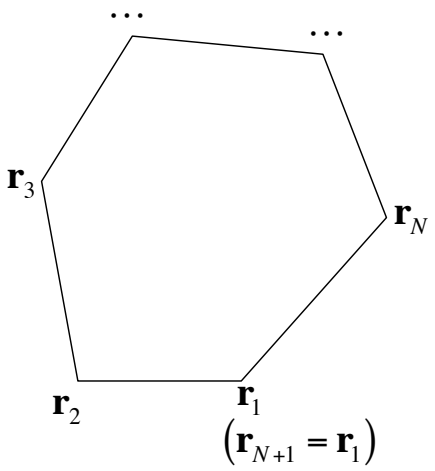



**Figure 20. Generic polygon.**

are numbered in a clockwise direction, signs on $I_{xx}$, $I_{yy}$, and $I_{xy}$ should be reversed so that $I_{xx}$ and $I_{yy}$ are always positive.

$$I_{xx} = \int_S y^2 dS = \sum_{i=1}^{N} \frac{1}{12} \left(X_i Y_{i+1} - X_{i+1} Y_i\right) \left(Y_i^2 + Y_i Y_{i+1} + Y_{i+1}^2\right) \tag{12}$$

$$I_{yy} = \int_S x^2 dS = \sum_{i=1}^{N} \frac{1}{12} \left(X_i Y_{i+1} - X_{i+1} Y_i\right) \left(X_i^2 + X_i X_{i+1} + X_{i+1}^2\right) \tag{13}$$

$$I_{xy} = -\int_S xy dS = -\sum_{i=1}^{N} \frac{1}{24} \left(X_i Y_{i+1} - X_{i+1} Y_i\right) \left(2X_i Y_i + X_i Y_{i+1} + X_{i+1} Y_i + 2X_{i+1} Y_{i+1}\right) \tag{14}$$

Moment of inertia tensor, $\mathbf{I}$, is constructed as shown in Eq. 15. Eigenvalues of $\mathbf{I}$, $I_{11}$ and $I_{22}$, become the moment of inertias along the principal directions, $\left(v_{1_x}, v_{1_y}\right)$ and $\left(v_{2_x}, v_{2_y}\right)$ respectively. These directions are the corresponding eigenvectors.

$$\mathbf{I} = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{bmatrix} = \begin{bmatrix} v_{1_x} & v_{2_x} \\ v_{1_y} & v_{2_y} \end{bmatrix} \begin{bmatrix} I_{11} & 0 \\ 0 & I_{22} \end{bmatrix} \begin{bmatrix} v_{1_x} & v_{2_x} \\ v_{1_y} & v_{2_y} \end{bmatrix}^T \tag{15}$$

Since the moment of inertias of a rectangle with width $w$, and height $h$ are expressed as Eq. 16, the aspect ratio, $h/w$, can be computed by Eq. 17.

$$\begin{aligned} I_{xx} &= \tfrac{1}{12} w h^3 \\ I_{yy} &= \tfrac{1}{12} w^3 h \end{aligned} \tag{16}$$

$$AR = \sqrt{\frac{I_{11}}{I_{22}}} \tag{17}$$

### Interior Condition

To determine whether a point is inside or outside the given polygon, a method based on potential theory is used. For a potential field, $\mathbf{q}$, the line integral around any closed curve depends on the number of singularities enclosed by the curve. Using this idea, a singularity is placed at the point, and the line integral around the given polygon is computed. This line integral is zero only if the point is outside the given polygon. If a

unit vortex is assumed for the potential field as shown in Eq. 18, the line integral around a polygon can be computed by using Eqs. 19 through 21.

$$\mathbf{q} = \nabla\phi = \nabla\frac{1}{2\pi\theta} \tag{18}$$

$$\oint_c \mathbf{q} \cdot d\mathbf{s} = \sum_{i=1}^{N} \int_{\mathbf{r}_i}^{\mathbf{r}_{i+1}} \mathbf{q} \cdot d\mathbf{s} \tag{19}$$

$$\int_{\mathbf{r}_i}^{\mathbf{r}_{i+1}} \mathbf{q} \cdot d\mathbf{s} = \frac{1}{2\pi} E \frac{2}{D} \left( \tan^{-1} \frac{B+2A}{D} - \tan^{-1} \frac{B}{D} \right) \tag{20}$$

$$\begin{aligned} A &= \|\mathbf{r}_{i+1} - \mathbf{r}_i\|^2 \\ B &= 2\mathbf{r}_i \cdot (\mathbf{r}_{i+1} - \mathbf{r}_i) \\ C &= \|\mathbf{r}_i\|^2 \\ D &= \sqrt{4AC - B^2} \\ E &= (\mathbf{r}_i \times \mathbf{r}_{i+1}) \end{aligned} \tag{21}$$

## References


[1]Erzberger, H. and Paielli, R. A., "Concept for Next Generation Air Traffic Control System," *Air Traffic Control Quarterly*, Vol. 10, No. 4, October 2002, pp. 355–378.

[2]Brinton, C., Krozel, J., and Capozzi, B., "Improved Taxi Prediction Algorithms for the Surface Management System," AIAA paper 2002-4857, Monterey, California, 5 - 8 August 2002.

[3]Windhorst, R. D. and Meyn, L. A., "The Airspace Concept Evaluation System Terminal Area Plant Model," AIAA paper 2007-6555, AIAA Modeling and Simulation Technologies Conference and Exhibit, Hilton Head, South Carolina, August 20-23 2007.

[4]Atkins, S., Carnoil, T., Feldman, M., and Neskovic, D., "Surface Management System (SMS) Adaptation Procedures Manual," NASA Contract NNA04BB31D.

[5]Mosaic ATM, Inc., *SODAA User's Guide version 1.7*, March 2008.

[6]Nolan, M. S., *Fundamentals of Air Traffic Control*, Brooks Cole, 4th ed., 2003.